1-1-2011

Electronic cash with anonymous user suspension

Man Ho Au
University of Wollongong, aau@uow.edu.au

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Yi Mu
University of Wollongong, ymu@uow.edu.au

Electronic cash with anonymous user suspension

Abstract

Electronic cash (E-cash) is the digital counterpart of cash payment. They allow users to spend anonymously unless they “double spend” their electronic coins. However, it is not possible to prevent users from misbehaving under some other subjective definitions of misbehavior, such as money laundering. One solution is to incorporate a trusted third party (TTP), which, upon complaint, uses its power to deanonymize the suspected user. This solution, known as fair e-cash, is not fully satisfactory since additional measure has to be taken to stop misbehaving users from further abusing the system after they have been identified. We present a e-cash system with anonymous user suspension, EC-AUS, which features an suspension manager (SM) that is capable of suspending the underlying user that participates in any suspicious transaction. Suspended users cannot participate in any transaction. The suspension is anonymous in the sense that no party, not even SM, can tell the identities of the suspended users nor link their past transactions. If they are found innocent later, their suspension can be revoked easily.

Keywords

anonymous, suspension, user, electronic, cash

Disciplines

Physical Sciences and Mathematics

Publication Details

Au, M., Susilo, W. & Mu, Y. (2011). Electronic cash with anonymous user suspension. Lecture Notes in Computer Science, 6812 (N/A), 172-188.

Electronic Cash with Anonymous User Suspension

Man Ho Au, Willy Susilo, and Yi Mu

Centre for Computer and Information Security Research
School of Computer Science and Software Engineering
University of Wollongong, Australia
{aau, wsusilo, ymu}@uow.edu.au

Abstract. Electronic cash (E-cash) is the digital counterpart of cash payment. They allow users to spend anonymously unless they "double spend" their electronic coins. However, it is not possible to prevent users from misbehaving under some other subjective definitions of misbehavior, such as money laundering. One solution is to incorporate a trusted third party (TTP), which, upon complaint, uses its power to deanonymize the suspected user. This solution, known as fair e-cash, is not fully satisfactory since additional measure has to be taken to stop misbehaving users from further abusing the system after they have been identified. We present a e-cash system with anonymous user suspension, EC-AUS, which features an suspension manager (SM) that is capable of suspending the underlying user that participates in any suspicious transaction. Suspended users cannot participate in any transaction. The suspension is anonymous in the sense that no party, not even SM, can tell the identities of the suspended users nor link their past transactions. If they are found innocent later, their suspension can be revoked easily.

1 Introduction

E-cash was introduced by David Chaum [18] as an electronic counterpart of physical money. Extensive research [19, 28, 24, 20, 7, 17, 23, 14] has been done on the subject since then. In an e-cash scheme, a user withdraws an electronic coin from the bank and the user can spend it to any merchant, who will deposit the coin back to the bank.

A secure and practical e-cash should possess three essential properties, namely, *anonymity*, *balance* and *exculpability*. *Anonymity* (also referred to as privacy), is a distinctive feature of cash payments offers a customer. It means that payments do not leak the customers' whereabouts, spending patterns or personal preferences. *Balance* means that no collusion of users and merchants together can deposit more than they withdraw *without* being detected. Finally, *exculpability* refers to the fact that honest spenders cannot be accused to have double-spent.

Too much privacy may cause problems in the regulatory levels since there is no way misbehaving users can be identified, let alone being punished. Spending the same electronic coin twice, also known as *double-spending*, is a prominent example of misbehavior. Existing e-cash schemes tackle this dilemma by incorporating mechanisms such that spending an electronic coin twice provides sufficient information for everyone to compute the user's identity.

Unfortunately, misbehavior cannot always be represented by mathematical relationships such as spending the same electronic coin twice. For instance, it is hard to define mathematically transactions for money laundering, illegal goods purchasing and blackmailing. Fair e-cash [15] addresses the issue by introducing an administrative party, called Open Authority (OA), which is capable of outputting the identity of a user participating in a transaction. This solution, however, does not stop the user from further abusing the system. The user can still spend all his other electronic coins after his identity is revealed. This gives the opportunity for the misbehaving user to transfer his money to some other accounts. In order to stop this, OA will have to open identities of all the transactions to check the flow of the money. The problem can be tackled using the technique of traceable signatures [27] in which the administrative party discloses some secret information, also known as tracing information, of a particular user, which, enables everyone to test if a spending belongs to that specific user. This property is sometimes known as coin

traceability [10]. The problem is, once the tracing information is disclosed, there is no way to restore the user's privacy even if he/she is found innocent later.

We think it is important to equip e-cash systems with anonymous user suspension in which users can be suspended without sacrificing their privacy. Suspended users are simply stopped from accessing the system, while their identities remain hidden. Law-enforcing agent can thus suspend users that participate in dubious transactions, investigate the case, and un-suspend the suspect if he/she is found innocent.

Our Contributions. We propose an *electronic cash with anonymous user suspension* (EC-AUS). We formalize the security model for such a system and prove that our construction is secure under this model. Furthermore, we also evaluate the performance of our system.

Paper Outline. In Section 2, we present preliminary information on the various cryptographic tools used in our construction. In Section 3, we formalize the syntax and security properties for EC-AUS. We present our construction and analyze the algorithmic complexity in Section 4. We discuss extensions and several other issues in Section 5 and conclude the paper in Section 6.

Related Work. Our EC-AUS is constructed based on the blacklisting technique from blacklistable anonymous authentication systems [29, 8]. Their idea can be summarized as follow. For each authentication, a user with secret key x provides the server with a unique value, called ticket t, which is b^x in some cyclic group $\mathbb{G}$ for a random nonce b. The server provides the user with a blacklist $\{(t_1, b_1), (t_2, b_2), \ldots, (t_n, b_n)\}$. In order to authenticate, the user proves to the server, in zero-knowledge, that $t_i \neq b_i^x$ for $i = 1$ to n and $t = b^x$. This assures the server that the authenticating user is not on the blacklist. If the server would like to blacklist this user later, the entry (t, b) is appended to the blacklist. If the Decisional Diffie-Hellman (DDH) Problem is hard in $\mathbb{G}$, the ticket t is unlinkable and thus user anonymity is preserved.

2 Preliminaries

In this section we define some notations and review cryptographic tools that we use as building blocks in our EC-AUS construction.

Notations. $|S|$ represents the cardinality of a set S. If S is a non-empty set, $a \in_R S$ means that a is drawn uniformly at random from S. If n is a positive integer, we write $[n]$ to mean the set $\{1, 2, \ldots, n\}$. If $s_1, s_2 \in \{0,1\}^*$, then $s_1 || s_2 \in \{0,1\}^*$ is the concatenation of binary strings s_1 and s_2. We say that a function $\mathsf{negl}(\lambda)$ is a negligible function [3], if for all polynomials $f(\lambda)$, for all sufficiently large λ, $\mathsf{negl}(\lambda) < 1/f(\lambda)$.

Bilinear Map. A pairing is a bilinear mapping from a pair of group elements to a group element. Specifically, let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic groups of prime order p. A function $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is said to be a pairing if it satisfies the following properties:

- (Bilinearity.) $\hat{e}(u^x, v^y) = \hat{e}(u, v)^{xy}$ for all $u, v \in \mathbb{G}_1$ and $x, y \in \mathbb{Z}_p$.
- (Non-Degeneracy.) $\hat{e}(g, g) \neq 1_{\mathbb{G}_2}$, the identity element of $\mathbb{G}_2$.
- (Efficient Computability.) $\hat{e}(u, v)$ is efficiently computable for all u, v.
- (Unique Representation.) All elements in $\mathbb{G}_1$, $\mathbb{G}_2$ have unique binary representation.

Proof of Knowledge. In a *Zero-Knowledge Proof of Knowledge (ZKPoK)* protocol [25], a prover convinces a verifier that some statement is true, while the verifier learns nothing except the validity of the statement. Σ-protocols are a special type of three-move ZKPoK protocols, which can be converted into non-interactive *Signature Proof of Knowledge (SPK)* schemes or simply signature schemes [26] that are secure in the *Random Oracle (RO)* Model [4]. Σ-protocols can be transformed to 4-move perfect zero-knowledge ZKPoK protocols [21]. They can also be transformed to 3-move

concurrent zero-knowledge protocol in the auxiliary string model using trapdoor commitment schemes [22].

We follow the notation introduced in [13]. For instance, $PK\{(x) : y = g^x\}$ denotes a Σ-protocol that proves the knowledge of $x \in \mathbb{Z}_p$ such that $y = g^x$ for some $y \in \mathbb{G}$. The values inside the parenthesis on the left of the colon denotes variables whose knowledge is to be proven, while values on the right of the colon except those inside the parenthesis denote publicly known value. We use $SPK\{(x) : y = g^x\}(M)$ to denote the transformation of the above Σ-protocol into signature of knowledge, which is secure in the random oracle model due to Fiat-Shamir heuristic. We employ several existing Σ-protocols as building blocks in our construction of EC-AUS. In particular, the ZKPoK of Knowledge and Inequalities of Discrete Logarithms due to Camenisch and Shoup [12].

BBS+ Signature. We briefly review the signature scheme proposed in [1], which is based on the schemes of [11] and [6]. This signature scheme also serves as building blocks in a number of cryptographic systems [2, 29, 9] and is referred to as BBS+ signature or credential signature.

Let g_0, g_1, g_2, ..., g_ℓ, $g_{\ell+1} \in \mathbb{G}_1$ be generators of $\mathbb{G}_1$. Let $\hat{e}$ be a bilinear map as discussed. Let $w = g_0^\gamma$ for some $\gamma \in_R \mathbb{Z}_p$. The public key of the signature scheme is $(g_0, \ldots, g_\ell, w, \hat{e})$, and the signing key is (γ).

A signature on messages $(m_1, \ldots, m_\ell)$ is a tuple (A, e, z), where e, z are random values in $\mathbb{Z}_p$ chosen by the signer such that $A = (g_0 g_1^{m_1} \cdots {g_\ell}^{m_\ell} g_{\ell+1}^z)^{\frac{1}{\gamma+e}}$. Such a signature can be verified by checking if

$$\hat{e}(A, w g_0^e) \stackrel{?}{=} \hat{e}(g_0 g_1^{m_1} \cdots {g_\ell}^{m_\ell} g_{\ell+1}^z, g_0).$$

It was proved in [1] that BBS+ is unforgeable under adaptively chosen message attack if the q-SDH assumption holds, where q is the number of signature queries, and that they also proposed a ZKPoK protocol which allows one to prove possession of message-signature pairs.

3 Security Definition

We present the syntax of EC-AUS, followed by the security properties that any EC-AUS construction must satisfy.

3.1 Syntax

The entities in EC-AUS are the *Suspension Manager (SM)*, *Bank (B)*, a set of *Merchants (M)* and a set of *users (U)*. EC-AUS consists of the following protocols/algorithms:

- $(\mathtt{bpk}, \mathtt{bsk}) \leftarrow \mathsf{BSetup}(1^\lambda)$. This algorithm is executed by the bank B to set up the system. On input of one or more security parameters (say, 1^λ), the algorithm outputs a pair consisting of public key bpk and private key bsk. B keeps bsk private and publishes bpk to the public. bpk is an implicit input to all the algorithms described below.
- $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{KeyGen}$. This algorithm is executed by the user or merchant to generate her key pairs. We assume there exists some kind of public key infrastructure that ensures the public key pk is properly certified and is a unique identifier for the user or merchant.
- $\{\mathsf{SUL} \leftarrow \mathsf{SSetup}\}$. SM maintains a suspended user list SUL which is available to all entities in the system and is empty initially.
- $\mathsf{AccEstablish}(\mathsf{B}(\mathtt{bsk}, \mathtt{pk}_\mathsf{U}), \mathsf{U}(\mathtt{pk}_\mathsf{U}, \mathtt{sk}_\mathsf{U}))$. This protocol is executed between B and a legitimate user U with public key $\mathtt{pk}_\mathsf{U}$ to establish an account. Upon successful completion of the protocol, the user obtains an account secret cred, which she keeps private to herself, and is thereby eligible for conducting transactions in the system.
- $\mathsf{Withdraw}(\mathsf{B}(\mathtt{bsk}, \mathsf{SUL}, \mathtt{pk}_\mathsf{U}), \mathsf{U}(\mathtt{cred}, \mathsf{SUL}, \mathtt{sk}_\mathsf{U}))$. This protocol is executed between B and a legitimate user U to withdraw an electronic coin. Upon successful completion of the protocol, the user obtains an electronic coin cn, which she keeps private to herself.

- $\mathsf{Spend}(\mathsf{M}(\mathtt{pk}_\mathsf{M}, \mathtt{sk}_\mathsf{M}, \mathsf{SUL}), \mathsf{U}(\mathtt{cred}, \mathtt{pk}_\mathsf{M}, \mathtt{cn}, \mathsf{SUL}))$. This protocol is executed between a merchant M with public key $\mathtt{pk}_\mathsf{M}$ and a legitimate user U to spend an electronic coin. Upon successful completion of the protocol, M accepts the coin and obtains a transcript $\mathtt{trans}$.
- $\mathsf{Deposit}(\mathsf{B}(\mathtt{bsk}, \mathtt{pk}_\mathsf{M}), \mathsf{M}(\mathtt{trans}, \mathtt{sk}_\mathsf{M}))$. This protocol is executed between B and a merchant M for the later to deposit an electronic coin. Upon successful completion of the protocol, B either accepts the request or outputs $\mathtt{pk}^*$, along with $\mathtt{trans}_1$, $\mathtt{trans}_2$, Π which serves as a proof that the party with public key $\mathtt{pk}^*$ has spent an electronic coin twice in transactions with transcripts $\mathtt{trans}_1$, $\mathtt{trans}_2$.
- $0/1 \leftarrow \mathsf{VerGuilt}(\mathtt{trans}_1, \mathtt{trans}_2, \mathtt{pk}^*, \Pi)$. Everyone can execute this algorithm to check if the party with public key $\mathtt{pk}^*$ indeed spent an electronic coin twice in transactions whose transcripts are $\mathtt{trans}_1$ and $\mathtt{trans}_2$.
- *Suspension.* This is a suite of three algorithms: $\varpi \leftarrow \mathsf{Extract}(\mathtt{trans})$, $\mathsf{SUL} \leftarrow \mathsf{Add}(\mathsf{SUL}', \varpi)$ and $\mathsf{SUL}' \leftarrow \mathsf{Remove}(\mathsf{SUL}, \varpi)$. These algorithms are executed by SM to suspend or un-suspend a user. On input of a Spend protocol transcript $\mathtt{trans}$, Extract extracts and returns a *ticket* ϖ from the transcript. The suspended user list SUL is a collection of tickets. On input of a SUL and a ticket, Add returns a new SUL that contains all the tickets in the input SUL' in addition to the input ticket. On the other hand, on input of SUL' and a ticket, Remove returns a new SUL that contains all the tickets in it, except the one(s) equivalent to the input ticket.
  When we say that a user Alice is suspended, we mean that there exists a Spend transaction between Alice and a merchant M with transcript $\mathtt{trans}$ such that the SM has invoked $\mathsf{Add}(\mathsf{SUL}, \mathsf{Extract}(\mathtt{trans}))$ and no $\mathsf{Remove}(\cdot, \mathsf{Extract}(\mathtt{trans}))$ has been invoked afterwards. If Alice is suspended, she cannot conduct Withdraw or Spend. We would like to stress that SM learns nothing about the identity of Alice, nor link any of Alice's past action. All SM does is to suspended an anonymous user that has participated in a spend that results in transcript $\mathtt{trans}$.

3.2 Security Requirements

We first describe various security properties that an EC-AUS construction must possess. Their formal definitions will be given in Appendix A.

- Balance. The bank B is assured that no collusion of users and merchants can deposit more than they withdraw *without being identified*. Consequently, any double spender in the system will be identified.
- Suspension-Correctness. B is assured to accept Withdraw, while Ms are assured to accept Spend, only from Us who are not suspended. On the other hand, honest users that are not currently suspended by SM can always conduct the above transaction with honest B or Ms.
- Anonymity. All that B, M and SM collude together can infer about the identity of a spender is whether that user is suspended at the time of protocol execution, and whether she is in possession of a valid electronic coin.
- Exculpability. An honest user will not be falsely accused of having spent an electronic coin twice. That is, B cannot output $(\mathtt{trans}_1, \mathtt{trans}_2, \mathtt{pk}^*, \Pi)$ such that $1 \leftarrow \mathsf{VerGuilt}(\mathtt{trans}_1, \mathtt{trans}_2, \mathtt{pk}^*, \Pi)$ even if B colludes with M and SM.

The trust placed on various parties regarding the security requirements are summarized in Table 1. The table is interpreted as follows. If Party A is to be assured Security Requirement B, he/she needs to trust the party with tick mark. For instance, users, bank and merchants need to trust that SM is honest for suspension-correctness to hold. Indeed, that is the only trust placed in our system. For instance, an honest user is guaranteed anonymity and exculpability even if the bank, merchant, suspension manager are malicious.

4 Our System

4.1 High Level Description

We provide a high level description of EC-AUS, which combines the technique of the e-cash scheme due to [2, 10] and the anonymous blacklisting technique from [29].

Table 1. Trust Relationship of various parties.

Party A	Security Requirements B	Bank	Suspension Manager	Merchant	User
Bank	Balance	N/A	×	×	×
User/Bank/Merchant	Suspension-Correctness	×	✓	×	×
User	Anonymity	×	×	×	×
User	Exculpability	×	×	×	×

The Setup. Let $\mathbb{G}$ be a cyclic group and g, h, h_0, h_1 are generators of $\mathbb{G}$. User and Merchant are equipped with key pairs of the form (g^x, x) where g is a generator of $\mathbb{G}$. The bank chooses a signature scheme and assume the key pair is (pk_{Sig}, sk_{Sig}). The public key of the bank is pk_{Sig}. The secret key is $\mathtt{sk}_{Sig}$. The suspension manager makes available an empty list, SUL.

Account Creation. User U with public key g^x creates an account with the bank B by submitting a value h^x to the bank, along with a proof-of-correctness.

Withdrawing an E-Coin. U first needs to show B he/she is not suspended. Both parties first obtain the current $\mathsf{SUL} = \{(t_1, b_1), (t_2, b_2), \ldots, (t_n, b_n)\}$ from suspension manager SM. U proves to B that using his secret key x, none of the relationships $t_i = b_i^x$ hold. B only issues U with an electronic coin if U is not suspended. An electronic coin for U is simply a signature $\sigma_{x,y}$ from B on values (x, y), where y is a random number unknown to B. $\sigma_{x,y}$ is issued in a "blind" way such that B learns nothing about x and y.

Spending an E-Coin. U needs to prove to M that he/she is not suspended before M would accept payment from U. Both parties first obtain the current $\mathsf{SUL} = \{(t_1, b_1), (t_2, b_2), \ldots, (t_n, b_n)\}$ from suspension manager SM. U and M agree on a unique transaction identifier R and a random value b. U then computes $S = {h_0}^y$, $T = h^x h_1^{Ry}$ and $t = b^x$ and proves the following facts.

1. U knows $\sigma_{x,y}$ which is a valid signature from B on values x, y.
2. S, T, t are formed correctly with respective to x and y.
3. $t_i \neq b_i^x$ for $i = 1$ to n.

Depositing an E-Coin. M submits (S, T, t, R) to B, along with the transcript `trans` of the spend operation. After checking the transcript, B checks if S is in its database. If yes, it is a coin that has been spent before. If not, it stores (S, T, t) in its database and credits M.

Dealing with Double-Spending. If B (S, T', t', R') is in its database, The public key of the double-spender can be computed as $(\frac{T^{R'}}{T^R})^{\frac{1}{R'-R}}$. Indeed, due to the soundness of the proof in the spend protocol, $T = h^x h_1^{Ry}$ and $T' = h^x h_1^{R'y}$. Thus $(\frac{T^{R'}}{T^R})^{\frac{1}{R'-R}} = ((h^x)^{R'-R})^{\frac{1}{R'-R}} = h^x = u$.

Suspension. To suspend a user, SM appends the value (b, t), in the protocol transcript of a spend operation, to SUL. Note that SM does not know the identity of the user being suspended; he just suspend the user that engage in this transaction. To un-suspend the user, SM removes that entry from SUL.

4.2 Construction Details

We now present our cryptographic construction of EC-AUS.

Parameters. Let λ be a sufficiently large security parameter. Let $(\mathbb{G}_1, \mathbb{G}_2)$ be a bilinear group pair such that $|\mathbb{G}_1| = |\mathbb{G}_2| = p$ for some prime p of λ bits. Also, let $\mathbb{G}$ be a group of order p where DDH Assumption holds. Let $g, g_0, g_1, g_2, g_3 \in \mathbb{G}_1$, $h, h_0, h_1 \in \mathbb{G}$ be generators of $\mathbb{G}_1$ and $\mathbb{G}$ respectively such that the relative discrete logarithm of the generators are unknown.[1] Let $H_0 : \{0,1\}^* \to \mathbb{G}$ and $H : \{0,1\}^* \to \mathbb{Z}_p$ be secure cryptographic hash functions, both of which will be modeled as random oracles.

BSetup, SSetup, KeyGen. The bank B randomly chooses $\gamma \in_R \mathbb{Z}_p$ and computes $w = g_0^\gamma$. The bank secret key is $\mathtt{bsk} = (\gamma)$ and the public key is $\mathtt{bpk} = (w)$.

[1] This can be done by setting the generators to be the output of a cryptographic hash function of some publicly known seeds. It is important for the users to verify this. For instance, knowledge of the discrete logarithm of h_1 to base h_0 would allow the bank to break the anonymity of the system.

The user U (resp. merchant M) randomly chooses $x \in_R \mathbb{Z}_p$ and computes $u = h^x$. The secret key is $\mathtt{sk} = (x)$ and the public key is $\mathtt{pk} = (u)$.

The suspension manager SM initializes the suspended user list SUL.

AccEstablish. User U sends her public key $\mathtt{pk}_\mathsf{U} = (u)$ to B, along with the following the zero-knowledge proof-of-knowledge $PK\{(x) : u = h^x\}$ to open an account in the bank. U stores the account secret $\mathtt{cred} = (x)$.

Withdraw.

1. U and B retrieve the current SUL from SM and parse SUL as $\{(t_1, b_1), \ldots, (t_n, b_n)\}$.
2. U initializes the request, claims to be the user with public key u who has already registered an account.
3. B sends a random challenge $m \in_R \mathbb{Z}_p$ to U.
4. U sends a pair (C, Π_1) to B, where $C = g_1^x g_2^y g_3^{z'} \in \mathbb{G}_1$ is a commitment of $x, y \in_R \mathbb{Z}_p$ using randomness z' and Π_1 is a signature proof of knowledge of
$$SPK_1 \left\{ (x, y, z') : C = g_1^x g_2^y g_3^{z'} \ \wedge \ u = h^x \ \left(\bigwedge_{i \in [n]} t_i \neq b_i^x \right) \right\} (m) \tag{1}$$
on challenge m, which proves that C is correctly formed.
5. The B returns `failure` if the verification of Π_1 returns `invalid`. Otherwise B sends U a tuple (A, e, z''), where $e, z'' \in_R \mathbb{Z}_p$ and $A = (g_0 C g_3^{z''})^{\frac{1}{e+\gamma}} \in \mathbb{G}_1$.
6. U computes $z = z' + z''$. She returns `failure` if $\hat{e}(A, w g_0^e) \neq \hat{e}(g_0 g_1^x g_2^y g_3^z, g_0)$. Otherwise she stores $\mathtt{cn} = (A, e, x, y, z)$ as her electronic coin.

Note that (A, e, z) is a BBS+ signature on values (x, y).

Spend. During an execution of this protocol between a user U and the merchant M, U's private input is her electronic coin $\mathtt{cn} = (A, e, x, y, z)$. Let R be the string that uniquely identifies this transaction. In particular, R includes the public key $\mathtt{pk}_\mathsf{M}$ of M, the version of SUL used and a random nonce `nonce`. When the protocol terminates, M outputs `success` or `failure`, indicating whether the payment is accepted. Both parties retrieve the current SUL from SM and parse SUL as $\{(t_1, b_1), \ldots, (t_n, b_n)\}$.

1. *(Challenge.)* M sends a random challenge $m \in_R \mathbb{Z}_p$to U.
2. *(Suspension Check.)* U returns `failure` if $t_i = b_i^x$ for some i (indicating that she is suspended). She proceeds otherwise.
3. *(Proof Generation.)* U returns to M a tuple (S, T, t, Π_2), where $S = h_0^y$, $T = u h_1^{yR}$, $t = b^x$ where $b = H_0(R)$. S is called the serial number of the coin while T is called a double-spending equation. The pair (S, T) allows the bank to identify the double spender. t is the ticket associated with the transaction which allows suspension. Finally, Π_2 is a signature proof of knowledge of:
$$SPK_2 \left\{ (A, e, x, y, z) : \begin{array}{c} \hat{e}(A, w g_0^e) = \hat{e}(g_0 g_1^x g_2^y g_3^z, g_0) \ \wedge \\ S = h_0^y \ \wedge \ T = h^x (h_1^R)^y \ \wedge \\ t = b^x \ \wedge \ \left(\bigwedge_{i \in [n]} t_i \neq b_i^x \right) \end{array} \right\} (m) \tag{2}$$
on the challenge m.
4. *(Proof Verification.)* M returns `failure` if the verification of Π_2 returns `invalid`. Otherwise it returns `success`.

Deposit. The merchant M submits the tuple (S, T, t, R, Π_2) to B. B first verifies if R contains a fresh nonce `nonce`, the public key $\mathtt{pk}_\mathsf{M}$ of M and obtains version of SUL used in this transaction. B then verifies Π_2. It runs through its database of spent coin, which is a list of tuples $(S_i, T_i, t_i, R_i, \Pi_{2,i})$. If S is not equal to any of the S_i, B credits M and appends (S, T, t, R, Π_2) to the list.

If R contains a reused nonce `nonce`, B outputs $\mathtt{pk}_\mathsf{M}$.

Otherwise, suppose there exists an entry $(S_j, T_j, t_j, R_j, \Pi_{2,j})$ for some index j in the list such that $S = S_j$, B computes $u^* = (\frac{T^{R_j}}{T_j^R})^{\frac{1}{R_j - R}}$ and outputs $\mathtt{pk}^* = u^*$, $\mathtt{trans}_1 = (S_j, T_j, t_j, R_j, \Pi_{2,j})$, $\mathtt{trans}_2 = (S, T, t, R, \Pi_2)$ and $\Pi = (\mathtt{trans}_1, \mathtt{trans}_2)$, indicating that u^* is the public key of the double spender.

VerGuilt. Since the computation of the identity of the double spender does not require bsk, everyone can verify the correctness of the bank's computation based on the two given transcripts.

Suspension. The three algorithms Extract, Add, Remove are all very simple and efficient. $\mathsf{Extract}\big(\langle S, T, t, R, \Pi_2\rangle\big)$ returns the ticket $\big(t, b = H_0(R)\big)$ in the input transcript. Of course, SM should also verify Π_2 to ensure that the transcript is valid. $\mathsf{Add}\big(\mathsf{SUL}, (t, b)\big)$ returns SUL', which is the same as the input SUL, with the input ticket (t, b) appended to it. $\mathsf{Remove}(\mathsf{SUL}, (t, b))$ returns SUL', which is the same as the input SUL, with all entries equal to the input ticket (t, b) dropped.

Formal security analysis of our construction is presented in Appendix A.

4.3 Efficiency Analysis

We analyze the efficiency of our construction in terms of both time and space/ communication complexities. Both complexities are linear in the size of SULfor Withdraw and Spend protocols. Below we analyze the most expensive operation, Spend, in our system.

Assume SUL contains n tickets. A proof Π_2 of SPK_2 consists of 2 $\mathbb{G}_1$ elements, n $\mathbb{G}$ elements and $2n + 10$ $\mathbb{Z}_p$ elements. The total communication complexity for a Spend protocol is thus $n + 1$ ℓ-bit strings, 5 $\mathbb{G}_1$ elements, n $\mathbb{G}$ elements and $2n + 10$ $\mathbb{Z}_p$ elements.

A breakdown of time complexity of the Spend protocol into the number of pairing operations and *multi-exponentiations (multi-EXPs)*[2] in various groups is shown in Table 2. Operations such as $\mathbb{G}$ addition and hashing have been omitted as computing them takes relatively insignificant time. Some preprocessing is possible at the user's side. In fact, all but $2n$ multi-EXPs in $\mathbb{G}$ can be precomputed by the user.

Table 2. Number of operations during a Spend protocol with a SUL of size n.

Operation	User w/o Preproc.	User w/ Preproc.	Merchant
$\mathbb{G}_1$ multi-EXP	6	0	3
$\mathbb{G}$ multi-EXP	$3n + 6$	$2n$	$2n + 3$
Pairing	1	0	1

5 Discussions

5.1 Incorporating Tracing Authority and Open Authority

Introduction of Open Authority (OA). It is relatively straightforward to introduce an Open Authority (OA) which is capable of revealing the public key of the user of any Spend transaction. One could simply require all users to verifiably encrypt [12] their public key g^x into ciphtertext C_x under the public key of the OA in the Spend transaction. This allows the OA to decrypt C_x and obtains the public key of the user participating in the Spend transaction.

Introduction of Tracing Authority (TA). We can also introduce TA in EC-AUS based on the idea of traceable signature [27]. Each user is issued a traceable signature signing key $k_{x,trace}$ from TA such that $k_{x,trace}$ is bind to the user public key g^x. All users are required to create a traceable signature σ_x using his key $k_{x,trace}$ in the Spend transaction. If the TA would like to trace the spending of a particular user, he/she reveal the tracing information of $k_{x,trace}$ so that every one can link the traceable signature σ_x from user with key $k_{x,trace}$ and thus link all his past actions.

[2] A multi-EXP computes the product of exponentiations faster than performing the exponentiations separately. We assume that one multi-EXP operation multiplies up to 3 exponentiations.

Three levels of anonymity revocation. The trio of TA, OA and SM provide a balance between users' privacy and accountability. For instance, when a suspicious transaction is identified, the law-enforcing agent can at once request a suspension from SM on the underlying user. Since SM reveals least information on the user, the threshold of issue could be fairy low. After preliminary investigations, law-enforcing agent could request TA to release the tracing information so that all transaction regarding the suspect can be linked and provide more information for the law-enforcing agent to make further investigation. Finally, he/she could request OA to reveal the identity of the spender for prosecution.

5.2 Managing the Size of SUL and the Bank's Database

Our system does not scale well with the size of the SUL. Thus, we assume that suspended users are eventually un-suspended if they are found innocent or their identity are revealed by the OA for prosecution. This would help keeping the size of SUL a minimum. Using practical parameters, modern computer handles a multi-EXP at around 2 ms. Realistically, EC-AUS would support SUL of size up to several thousands.

Another issue is the requirement that the bank has to keep record of all the electronic coins deposited. One solution is to limit the lifetime of the user account as well as the coins. Users are required to establish a new account and have their electronic coins re-issued at the end of the period. The account and coins are made valid only for a specific period of time, say, a month, several months or a year which offers a trade-off between database size and the frequency of the re-issue. Thus, the bank only needs to record spent coins of the current time period.

6 Conclusion

We presented EC-AUS, an electronic cash system with anonymous user suspension. Since suspended users remain anonymous, misbehavior can be judged subjectively and imposed with less cation. We also discuss how to limit the size of the bank's storage. We believe the ability to suspend users while maintaining their anonymity is a worthwhile endeavor. We left it as an open problem of constructing schemes whose complexities is independent to the size of SUL.

References

1. M. H. Au, W. Susilo, and Y. Mu. Constant-Size Dynamic k-TAA. In R. D. Prisco and M. Yung, editors, *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2006.
2. M. H. Au, W. Susilo, and Y. Mu. Practical Compact E-Cash. In *ACISP*, pages 431–445, 2007.
3. M. Bellare. A Note on Negligible Functions. *J. Cryptology*, 15(4):271–284, 2002.
4. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
5. D. Boneh and X. Boyen. Short Signatures without Random Oracles. In *EUROCRYPT*, pages 56–73, 2004.
6. D. Boneh, X. Boyen, and H. Shacham. Short Group Signatures. In *CRYPTO*, volume 3152 of *LNCS*, pages 41–55, 2004.
7. S. Brands. Untraceable Off-line Cash in Wallets with Observers (Extended Abstract). In *CRYPTO*, pages 302–318, 1993.
8. E. Brickell and J. Li. Enhanced Privacy ID: A Direct Anonymous Attestation Scheme with Enhanced Revocation Capabilities. In *WPES*, pages 21–30, 2007.
9. J. Camenisch, M. Dubovitskaya, and G. Neven. Oblivious Transfer with Access Control. In E. Al-Shaer, S. Jha, and A. D. Keromytis, editors, *ACM Conference on Computer and Communications Security*, pages 131–140. ACM, 2009.
10. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact E-Cash. In *EUROCRYPT*, pages 302–321, 2005.
11. J. Camenisch and A. Lysyanskaya. A Signature Scheme with Efficient Protocols. In *SCN*, pages 268–289, 2002.

12. J. Camenisch and V. Shoup. Practical Verifiable Encryption and Decryption of Discrete Logarithms. In *CRYPTO'03*, volume 2729 of *LNCS*, pages 126–144, 2003.
13. J. Camenisch and M. Stadler. Efficient Group Signature Schemes for Large Groups (Extended Abstract). In *CRYPTO*, pages 410–424, 1997.
14. S. Canard and A. Gouget. Divisible E-Cash Systems can be Truly Anonymous. In *EUROCRYPT*, pages 482–497, 2007.
15. S. Canard and J. Traoré. On Fair E-cash Systems Based on Group Signature Schemes. In *ACISP*, pages 237–248, 2003.
16. R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. Cryptology ePrint Archive, Report 2000/067, 2000. http://eprint.iacr.org/.
17. A. H. Chan, Y. Frankel, and Y. Tsiounis. Easy Come - Easy Go Divisible Cash. In *EUROCRYPT*, pages 561–575, 1998.
18. D. Chaum. Blind Signatures for Untraceable Payments. In *Advances in Cryptology: Proceedings of CRYPTO '82*, pages 199–203. Plenum, New York, 1983.
19. D. Chaum, A. Fiat, and M. Naor. Untraceable Electronic Cash. In *CRYPTO*, pages 319–327, 1988.
20. D. Chaum and T. P. Pedersen. Transferred Cash Grows in Size. In *EUROCRYPT*, pages 390–407, 1992.
21. R. Cramer, I. Damgård, and P. D. MacKenzie. Efficient Zero-Knowledge Proofs of Knowledge without Intractability Assumptions. In *PKC'00*, volume 1751 of *LNCS*, pages 354–373, 2000.
22. I. Damgård. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In *EUROCRYPT'00*, volume 1807 of *LNCS*, pages 418–430, 2000.
23. T. Eng and T. Okamoto. Single-Term Divisible Electronic Coins. In *EUROCRYPT*, pages 306–319, 1994.
24. M. K. Franklin and M. Yung. Secure and Efficient Off-Line Digital Money (Extended Abstract). In *ICALP*, pages 265–276, 1993.
25. S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract). In *STOC*, pages 291–304, 1985.
26. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
27. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable Signatures. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 571–589. Springer, 2004.
28. T. Okamoto and K. Ohta. Universal Electronic Cash. In *CRYPTO*, pages 324–337, 1991.
29. P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith. Blacklistable Anonymous Credentials: Blocking Misbehaving Users without TTPs. In *ACMCCS 2007*, pages 72–81, 2007.

A Formal Security Analysis

A.1 Security Model

We use a simulation-based approach to define security of EC-AUS formally. We would like to remark that the definition we give do not entail all formalities necessary to fit into the universal composability framework [16]; our goal here is to prove security of our construction. Our model is static in the sense that the adversary could not corrupt honest users and merchants during the execution of the system.

We summarize the ideas of the model. The players in the system are the suspension manager SM, the bank B, a set of users Us and a set of merchants Ms. In the real world there are a number of players who communicate via cryptographic protocols. Then there is an adversary $\mathcal{A}$, who controls the dishonest players in the system. We define an entity called environment, $\mathcal{E}$, who provides the inputs to the players and receives their outputs. $\mathcal{E}$ also interacts freely with the adversary $\mathcal{A}$.

In the ideal world, we have the same players. However, they do not communicate directly. Rather, there exists a trusted party $\mathcal{T}$ who is responsible for all handling operations for all players. Specifically, $\mathcal{T}$ computes the outputs of the players from their inputs, that is, applies the functionality that the cryptographic protocols are supposed to realize. The environment $\mathcal{E}$ again provides the inputs to, and receives the outputs from, the players, and interacts arbitrarily with $\mathcal{A}$ who controls the dishonest players.

The ideal world. First we define the ideal world specification of EC-AUS. Communication between a player and the trusted party $\mathcal{T}$ is not anonymous. Ideal world EC-AUS supports the following operations. These operations are scheduled according to the environment $\mathcal{E}$'s wish. Each call to the operation is assigned a unique identifier $\mathtt{tid}$. We also describe the behavior of $\mathcal{T}$ based on the inputs of the ideal world players for the following operations. The SUL in the ideal world is a list of $\mathtt{tid}$ of Spend operation.

- $\mathtt{tid}_0 \leftarrow \mathsf{SSetup/BSetup/KeyGen}(\mathcal{HP}, \mathcal{AP})$. The system begin when $\mathcal{E}$ invokes this operation which specified the set of honest players $\mathcal{HP}$ and dishonest players $\mathcal{AP}$. This must be the first operation in the schedule and can only be called once.
- $\mathtt{tid}_A \leftarrow \mathsf{AccEstablish}(i)$. $\mathcal{E}$ instructs user U_i to establish an account with bank B. U_i sends a request to $\mathcal{T}$, $\mathcal{T}$ checks U_i has never established an account before and informs B that U_i would like to establish an account. B returns $\mathtt{accept/reject}$ to $\mathcal{T}$ and $\mathcal{T}$ forward it to U_i. Both U_i and B output $(\mathtt{tid}_A, \mathtt{accept/reject})$ to $\mathcal{E}$ individually.
- $\mathtt{tid}_W \leftarrow \mathsf{Withdraw}(i)$. $\mathcal{E}$ instructs user U_i to withdraw an electronic coin from B. U_i sends a request to $\mathcal{T}$, $\mathcal{T}$ requests the current version of SUL from SM and forward SUL to U_i, along with a check result that indicate if U_i is suspended or not. U_i replies to $\mathcal{T}$ if he/she chooses to proceed or not. $\mathcal{T}$ requests the same version of SUL from SM, check if U_i has ever participated in the Spend specified in this SUL and forwards SUL to B, a bit indicating if U_i is suspended and the request that U_i would like to withdraw an electronic coin. B returns $\mathtt{accept/reject}$ to $\mathcal{T}$ and $\mathcal{T}$ forwards it to U_i. If B returns $\mathtt{accept}$, $\mathcal{T}$ stores $\mathtt{tid}_W$ as U_i's un-spent coin. Both U_i and B output $(\mathtt{tid}_W, \mathtt{accept/reject})$ to $\mathcal{E}$ individually.
- $\mathtt{tid}_S \leftarrow \mathsf{Spend}(i, \mathtt{tid}_W, j)$. $\mathcal{E}$ instructs user U_i to spend the electronic coin he/she obtains in transaction $\mathtt{tid}_W$ to merchant M_j. U_i sends a request to $\mathcal{T}$, $\mathcal{T}$ requests the current version of SUL from SM and forward SUL to U_i, along with a check result that indicate if U_i is suspended or not as well as whether $\mathtt{tid}_W$ corresponds to an un-spent coin of U_i. U_i replies to $\mathcal{T}$ if he/she chooses to proceed or not. $\mathcal{T}$ requests the same version of SUL from SM, check if U_i is suspended and forwards SUL to M_j, the request that an anonymous user that would like to spend a coin to M_j, and whether this user is suspended or not and whether U_i is having a valid coin (valid means $\mathtt{tid}_W$ corresponds to a Withdraw that U_i participated in, it might be a spent-coin though). M_j returns $\mathtt{accept/reject}$ to $\mathcal{T}$ and $\mathcal{T}$ forwards it to U_i. If M_j returns $\mathtt{accept}$, $\mathcal{T}$ marked $\mathtt{tid}_W$ as U_i's spent coin. Both U_i and M_j output $(\mathtt{tid}_S, \mathtt{accept/reject})$ to $\mathcal{E}$ individually.
- $\mathtt{tid}_D \leftarrow \mathsf{Deposit}(j, \mathtt{tid}_S)$. $\mathcal{E}$ instructs user M_j to deposit the electronic coin he/she obtains in transaction $\mathtt{tid}_S$. M_j sends a request to $\mathcal{T}$, $\mathcal{T}$ requests the version of SUL used during Spend of $\mathtt{tid}_S$ from SM and forward SUL to B, along with a check result that indicate if $\mathtt{tid}_S$ corresponds to a $\mathsf{Spend}i, \mathtt{tid}_W, j$ that results in M_j outputting $\mathtt{accept}$ and that U_i is not suspended based on SUL. Next $\mathcal{T}$ also informs B if $\mathtt{tid}_W$ corresponds to a deposited-coin. If yes, $\mathcal{T}$ gives B an identity, U_i or M_j, indicating if the coin is spent twice by U_i or deposited twice by M_j. If not, $\mathcal{T}$ marks $\mathtt{tid}_W$ as a deposited coin from M_j. Both B and M_j output $(\mathtt{tid}_D, \mathtt{accept/reject}(\mathsf{U}_i/\mathsf{M}_j))$ to $\mathcal{E}$ individually.
- $\mathtt{tid}_V \leftarrow \mathsf{VerGuilt}(P, \mathtt{tid}_D, \mathsf{U}_i/\mathsf{M}_j)$. $\mathcal{E}$ instructs any player P to query if B outputs a correct double-spender in transaction $\mathtt{tid}_D$. $\mathcal{T}$ replies with a bit, indicating if the correct double-spender is outputted in $\mathtt{tid}_D$.
- $\mathtt{tid}_{Sus} \leftarrow \mathit{Suspend}(\mathtt{tid}_{\S})$. $\mathcal{E}$ instructs SM to add the Spend identified by $\mathtt{tid}_S$ to SUL.
- $\mathtt{tid}_{Un-Sus} \leftarrow \mathit{Un-Suspend}(\mathtt{tid}_{\S})$. $\mathcal{E}$ instructs SM to removes the entry $\mathtt{tid}_S$ from SUL.

Ideal world EC-AUS provides all the desired security properties. Firstly, all Spend transaction are anonymous. $\mathcal{T}$ only informs and M a certain anonymous user would like to spend an e-coin. Thus, anonymity and exculpability is guaranteed. Secondly, $\mathcal{T}$ verifies if validity of the user during Withdraw, Spend and Depositand thus balance is assured. Finally, $\mathcal{T}$ consults SM for SUL and checks if the underlying user is suspended for the B and M and thus suspension-correctness is attained.

Next, we define a cryptographic EC-AUS which also supports the above eight types of transaction. Since there is no trusted party $\mathcal{T}$, the functionalities are realized through cryptographic means. Below we highlight the difference.

- $\mathtt{tid}_0 \leftarrow \mathsf{SSetup/BSetup/KeyGen}(\mathcal{HP}, \mathcal{AP})$. SM, B, Us and Ms invokes the respective algorithms SSetup, BSetup and KeyGen.
- $\mathtt{tid}_A \leftarrow \mathsf{AccEstablish}(i)$. U_i and B obtains the current version of SUL from SM individually and engage in the AccEstablish protocol.
- $\mathtt{tid}_W \leftarrow \mathsf{Withdraw}(i).\mathsf{U}_i$ and B obtains the current version of SUL from SM individually and engage in the Withdraw protocol.
- $\mathtt{tid}_S \leftarrow \mathsf{Spend}(i, \mathtt{tid}_W, j)$. U_i and M obtains the current version of SUL from SM individually and engage in the Spend protocol.
- $\mathtt{tid}_D \leftarrow \mathsf{Deposit}(j, \mathtt{tid}_S)$. M_j and B engage in the Deposit protocol in which B obtains from SM the version of SUL used in the Spend protocol identified by $\mathtt{tid}_S$.
- $\mathtt{tid}_V \leftarrow \mathsf{VerGuilt}(P, \mathtt{tid}_D, \mathsf{U}_i/\mathsf{M}_j)$. P interacts with B who proves to P the identity of the double-spender is correctly computted.

Informally speaking, a cryptographic system is secure if for every real world adversary $\mathcal{A}$ and every environment $\mathcal{E}$, there exists an ideal world adversary $\mathcal{S}$ controlling the same players in the ideal world as $\mathcal{A}$ does in the real world such that, $\mathcal{E}$ cannot tell whether it is running in the real world interacting with $\mathcal{A}$ or it is running in the ideal world interacting with $\mathcal{S}$ which has blackbox access to $\mathcal{A}$. The rationale is that since by default the ideal world EC-AUS is secure, and the real world EC-AUS is indistinguishable to the ideal world EC-AUS, the real world EC-AUS is also secure. Formally, we define it in Definition 1.

Definition 1 (Security). *Let* $\mathbf{Real}_{\mathcal{E},\mathcal{A}}(\lambda)$ *(resp.* $\mathbf{Ideal}_{\mathcal{E},\mathcal{S}_\mathcal{A}}(\lambda)$ *) be the probability that* $\mathcal{E}$ *outputs 1 when run in the real world (resp. ideal world) with adversary* $\mathcal{A}$ *(resp.* $\mathcal{S}$ *having blackbox access to* $\mathcal{A}$*). A EC-AUS construction is secure if*

$$|\mathbf{Real}_{\mathcal{E},\mathcal{A}}(\lambda) - \mathbf{Ideal}_{\mathcal{E},\mathcal{S}_\mathcal{A}}(\lambda)| = \mathsf{negl}(\lambda)$$

for every PPT algorithms $\mathcal{E}$*,* $\mathcal{A}$*.*

A.2 Security Analysis

The security of our EC-AUS construction depends on the following two assumptions:

Definition 2 (DDH). *The Decisional Diffie-Hellman (DDH) problem in group* $\mathbb{G}$ *is defined as follows: On input of a quadruple* $(g, g^a, g^b, g^c) \in \mathbb{G}^4$*, output 1 if* $c = ab$ *and 0 otherwise. We say that the DDH assumption holds if no probabilistic polynomial time (PPT) algorithm has non-negligible advantage over random guessing in solving the DDH problem.*

Definition 3 (q-SDH). *The q-Strong Diffie-Hellman (q-SDH) problem in* $\mathbb{G}$ *is defined as follows: On input of a* $(q+1)$*-tuple* $(g,\ g^x,\ g^{x^2},\ \ldots,\ g^{x^q}) \in \mathbb{G}$*, output a pair* $(A, e) \in \mathbb{G} \times \mathbb{Z}_p$ *such that* $A^{(x+e)} = g$ *where* $|\mathbb{G}| = p$*. We say that the q-SDH assumption holds if no PPT algorithm has non-negligible advantage in solving the q-SDH problem.*

The q-SDH assumption was introduced by Boneh and Boyen [5] when they proposed a new short signature. They derived a lower bound on any generic algorithm that solves the q-SDH problem.

Regarding the security of EC-AUS, we have the following theorem.

Theorem 1. *If the q-SDH assumption holds in* $\mathbb{G}_1$ *and the DDH assumption holds in* $\mathbb{G}$*, our construction of EC-AUS satisfies Definition 1 in the random oracle model.*

Proof of Theorem 1 is done by showing the indistinguishability between adversary actions in the real world and the ideal world. The idea of the proof is that, given a real world adversary $\mathcal{A}$, we show how to construct an ideal world adversary $\mathcal{S}_{\mathcal{A}}$[3] such that no environment $\mathcal{E}$ can distinguish whether it is interacting with $\mathcal{A}$ or $\mathcal{S}$. The proof is divided into three cases according to the subset of players controlled by $\mathcal{A}$. In the first case, $\mathcal{A}$ controls the SM, a subset of merchants and users. This covers the security requirement of balance. In the second case, $\mathcal{A}$ controls SM, the bank, a subset of merchants and users. This covers the security requirement of exculpability and anonymity. In the third case, $\mathcal{A}$ control a subset of merchants and users. This covers the security requirement of suspension-correctness. We would like to remark that the three cases are orthogonal because on one hand, $\mathcal{S}$ has to represent all honest players to $\mathcal{A}$, while on the other hand $\mathcal{S}$ has to represent all dishonest players to $\mathcal{E}$. Thus, an adversary $\mathcal{A}$ controlling fewer parties does not necessarily makes the construction of $\mathcal{S}$ easier. We complete the proof with the following three lemmas.

Lemma 1. *For any environment $\mathcal{E}$ and real world adversaries $\mathcal{A}$ controlling the SM, some subsets of merchants and users, there exists an ideal world simulator $\mathcal{S}_{\mathcal{A}}$ such that*

$$|\mathbf{Real}_{\mathcal{E},\mathcal{A}}(\lambda) - \mathbf{Ideal}_{\mathcal{E},\mathcal{S}_{\mathcal{A}}}(\lambda)| = \mathsf{negl}(\lambda)$$

Proof. (Sketch) We construct $\mathcal{S}$ as follow. On one hand, $\mathcal{S}$ represents the honest merchants Ms, users Us and the bank B to $\mathcal{A}$ while on the other hand, $\mathcal{S}$ represents the dishonest Us, Ms and SM to $\mathcal{T}$ as well as $\mathcal{E}$ based on the actions from $\mathcal{A}$. $\mathcal{S}$ forwards all the messages between between $\mathcal{E}$ and $\mathcal{A}$. Next, for all AccEstablish involving a dishonest user, $\mathcal{S}$, playing the role of B, extracts the secret x from $\mathcal{A}$ and uses x as an index for the underlying dishonest user. It then represents that dishonest user to $\mathcal{T}$ and initiates an AccEstablish request. For all Withdraw involving a dishonest user, $\mathcal{S}$ extracts the values (x, y, z') from Eq.1. For all Spend events involving a dishonest user, $\mathcal{S}$, playing the role of an honest merchant, runs through its list of (x, y, z) extracted and locate the user by testing if $S = h_0^y$ and $t = b^x$ and locate the `tid` of the corresponding Withdraw event in the ideal world. It then represents that dishonest user to $\mathcal{T}$ and initiates a Spend request.

$\mathcal{S}$'s behavior in the view of $\mathcal{E}$ is exactly the same as $\mathcal{A}$ would provide, except in the case when $\mathcal{S}$ cannot extract the values from $\mathcal{A}$, or the extracted values do not matches with previously extracted one. This represents $\mathcal{A}$ is able to break the soundness of the various zero-knowledge proves, which happens with negligible probability under the q-SDH Assumption.

Lemma 2. *For any environment $\mathcal{E}$ and any real world adversaries $\mathcal{A}$ controlling SM, the bank, some subsets of merchants and users, there exists an ideal world simulator $\mathcal{S}_{\mathcal{A}}$ such that*

$$|\mathbf{Real}_{\mathcal{E},\mathcal{A}}(\lambda) - \mathbf{Ideal}_{\mathcal{E},\mathcal{S}_{\mathcal{A}}}(\lambda)| = \mathsf{negl}(\lambda)$$

Proof. (Sketch) Construction of such $\mathcal{S}$ is straightforward. $\mathcal{S}$ forwards all the messages between $\mathcal{E}$ and $\mathcal{A}$. For all events when $\mathcal{S}$ has to represent an honest user, $\mathcal{S}$ employs the zero-knowledge simulator to simulate the proofs using a random and different (x, y). $\mathcal{S}$'s behavior in the view of $\mathcal{E}$ is exactly the same as $\mathcal{A}$ would provide, except in the case when $\mathcal{A}$ is able to break the zero-knowledgeness. This happens with negligible probability under the DDH Assumption.

Lemma 3. *For any environment $\mathcal{E}$ and any real world adversaries $\mathcal{A}$ controlling some subsets of merchants and users, there exists an ideal world simulator $\mathcal{S}_{\mathcal{A}}$ such that*

$$|\mathbf{Real}_{\mathcal{E},\mathcal{A}}(\lambda) - \mathbf{Ideal}_{\mathcal{E},\mathcal{S}_{\mathcal{A}}}(\lambda)| = \mathsf{negl}(\lambda)$$

Proof. (Sketch) Construction of such $\mathcal{S}$ is straightforward. Again, $\mathcal{S}$ forwards all the messages between $\mathcal{E}$ and $\mathcal{A}$. For all events when $\mathcal{S}$ has to deal with dishonest user, $\mathcal{S}$ extracts the user secret x from the zero-knowledge proofs. $\mathcal{S}$'s behavior in the view of $\mathcal{E}$ is exactly the same as $\mathcal{A}$ would provide, except in the case when $\mathcal{S}$ cannot extract the values from $\mathcal{A}$, or the extracted values do not matches with previously extracted one. This represents $\mathcal{A}$ is able to break the soundness of the various zero-knowledge proves, which happens with negligible probability under the q-SDH Assumption.

[3] The subscript $\mathcal{A}$ is used to emphasis that $\mathcal{S}$ is given blackbox access to $\mathcal{A}$.